MODELLING OF THE BREAKDOWN VOLTAGE OF SOLID INSULATING MATERIALS USING SOFT COMPUTING TECHNIQUES

A project report submitted in partial fulfillment of the requirements for the degree of
Bachelor of Technology in Electrical Engineering

By

SUBHAPARNA GHOSH
ANDY B. SWER

Under the guidance of
PROF. S. MOHANTY

NATIONAL INSTITUTE OF TECHNOLOGY Rourkela
Rourkela -769008, Orissa

ACKNOWLEDGEMENT

We would like to express our gratitude towards all the people who have contributed their precious time and effort to help us .without whom it would not have been possible for us to understand and complete this project.

We would like to thank Prof. *S, MOHANTY* ,Department of Electrical Engineering my supervisor for his guidance ,support ,motivation and encouragement through out the period this work was carried out .His readiness for consultation at all times ,his educative comments ,his concern and assistance even with practical things have been invaluable.

We are grateful to *PROF. B.D SUBUDHI ,professor and head ,Dept. of Electrical Engineering* for providing necessary facilities in the department

SUBHAPARNA GHOSH(10602002)

ANDY B. SWER(10602054)

NATIONAL INSTITUTE OF TECHNOLOGY Rourkela
Rourkela-769008, Orissa

CERTIFICATE

This is to certify that the Project entitled "**MODELLING OF THE BREAKDOWN VOLTAGE OF SOLID INSULATING MATERIALS USING SOFT COMPUTING TECHNIQUES"** submitted by sri Subhaparna ghosh and Andy B. Swer has in partial fulfillment of the requirements for the award of **Bachelor of Technology** in **Electrical Engineering** at **National Institute of Technology ,Rourkela** is an authentic work carried out by them under my supervision and guidance

Place : NIT Rourkela
DATE:

(Prof . S. Mohanty)
Department of Electrical Engineering
NIT Rourkela

Chapter 1

INTRODUCTION

Abstract

Objective

Background

ABSTRACT

The aim of the project is to use Soft Computing Techniques (SCT) in order to model the breakdown voltage of solid insulating materials. Since the breakdown voltage behaviour is non-linear, it can be best modeled using SCT such as Artificial Neural Network (ANN), Radial Basis Function (RBF) Network, Fuzzy Logic (FL) Techniques etc. In order to obtain the experimental data on the breakdown voltage, experiments are conducted under AC and DC conditions and then all the SCT model are applied on it.

OBJECTIVE

The prediction of the breakdown voltage of solid insulating materials is indeed a challenging task. Hence the best way to go about it is by resorting to SCT in order to model and predict the breakdown voltage.

BACKGROUND

In order to carry out the project work, text books on ANN, RBF and FL are being studied. Since program is to be written in the modeling work, the MATLAB package is also being familiarized

CHAPTER 2

Why artificial neural networks?

Neural network: a brief introduction

Why artificial neural networks?

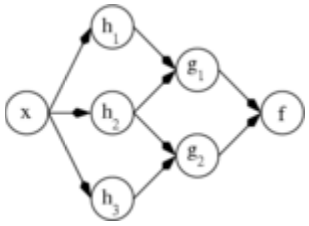
An **artificial neural network (ANN)**, usually called "neural network" (NN), is a mathematical model or computational model that tries and simulate the structure aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a interconnected approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the initial learning phase. Modern neural networks are non-linear statistical data modeling equipments. They are usually used to model complex relationships between inputs and outputs or to find any patterns in data.
In the case of modeling of breakdown characteristics of solid insulating materials the relationship between the input and the output is unknown hence modeling techniques such as ANN is used to predict breakdown voltage of any material.

Neural Networks: A brief introduction

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially mathematical models defining a function $f : X \rightarrow Y$. Each type of ANN model corresponds to a class of such functions.

The network in artificial neural network

he word *network* in the term 'artificial neural network' arises because the function *f*(*x*) is defined as a composition of other functions $g_i(x)$, which can further be defined as a mixture of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where *K* (commonly referred to as the activation function) is some predefined function. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.



ANN dependency graph

This figure tells us such a decomposition of *f*, with dependencies between variables indicated by arrows. These can be interpreted in two ways.

The first view is called the functional view: the input *x* is transformed into a 3-dimensional vector *h*, which is then transformed into a 2-d vector *g*, which is finally transformed into *f*. This view is most commonly encountered in the context of optimization.

The second view is called the probabilistic view: the random variable *F* = *f*(*G*) depends upon the random variable *G* = *g*(*H*), which depends upon *H* =

h(*X*), which depends on the random variable *X*. This view is most commonly encountered in the case of graphical models.

The two views are almost equivalent. In either case, for this particular network architecture, the components of individual layers are independent of each other (e.g., the components of *g* are independent of each other given their input *h*). This naturally establishes a degree of parallelism in the implementation. Networks such as the previous one are called feed forward, because their graph is a directed *acyclic graph*. Networks with cycles are called *recurrent*.

Learning

What has attracted interest in neural networks is the possibility of *learning*. Given a specific *task* to solve, and a *class* of functions *F*, learning means using a set of *observations* to find $f^* \in F$ which solves the task in some *optimal* sense.

This entails defining a cost function $C : F \to \mathbb{R}$ such that, for the optimal solution f^*, $C(f^*) \leq C(f) \forall f \in F$.

The cost function *C* is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost or error.

For applications where the solution depends on some data, the cost must necessarily be a *function of the observations*, otherwise we would not be able to model anything related to the data. It is frequently defined as a statistic to which only approx. can be made. As a simple example consider the problem of finding the model *f* which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (*x*,*y*) drawn from some distribution $\mathcal{D}$. In practical situations we would only have *N* samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire data set.

Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task.

1. Supervised learning

2. Unsupervised learning

3. Reinforced learning

Learning algorithms (training)

Training a neural network model essentially means selecting one model from the set of allowed models that minimizes the error. There are many algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and estimation.
Most of the algorithms used in training artificial neural networks employ some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Employing artificial neural networks

- Choice of model: This will depend on the data representation and the application. Overly complex models tend to lead to problems with learning.
- Learning algorithm: There are numerous compromise between learning algorithms. Almost any algorithm will work well with the *correct parameters* for training on a particular fixed set of data. However selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.
- Robustness: If the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

Types of Neural Networks

1. **Feed forward networks** : Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs directly with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up .
2. **Feed back networks:** Feedback networks can have signals moving in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated in some cases. Feedback networks are dynamic; their 'state' changes continuously until they reach an equilibrium value. They remain at the equilibrium point until the input changes and a new equilibrium is found. Feedback architectures are also known as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations
3. **Radial basis function network:** Radial Basis Functions are powerful techniques for interpolation in multidimensional space. A RBF is a function which has built into a distance criterion with respect to a center. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in Multi-Layer Perceptrons. RBF networks have two layers of processing: In the first, input is mapped onto each RBF in the 'hidden' layer. The RBF chosen is usually a Gaussian. In regression problems the output layer is then a linear combination of hidden layer values representing mean predicted output. The interpretation of this output layer value is the same as a regression model in statistics. In classification problems the output layer is typically a sigmoid function of a linear combination of hidden layer values, representing a posterior probability. Performance in both cases is often improved by shrinkage techniques, known as ridge regression in classical statistics and known to correspond to a prior belief in small parameter values (and therefore smooth output functions) in a Bayesian framework.

4. Perceptron network: The perceptron is a binary categorizer that maps its input *x* (a real-valued vector) to an output value *f*(*x*) (a single binary value) throughout the matrix.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

where *w* is a vector of real-valued weights and $w \cdot x$ is the dot product (which computes a weighted sum). *b* is the 'bias', a constant term that does not depend on any given input value.

The value of *f*(*x*) (0 or 1) is used to define *x* as either a positive or a negative instance, in the case of a binary classification problem. The bias can be defined as offsetting the activation function, or giving the output neuron a "base" level of activity. If *b* is negative, then the weighted combination of inputs must produce a positive value greater than | *b* | in order to push the classifier neuron over the 0 threshold. the bias alters the position of the decision boundary.

Since the inputs are fed directly to the output unit via the weighted connections, the perceptron can be considered the simplest kind of feed-forward neural network.

CHAPTER 3

Insulators: a brief overview

INSULATORS

An **insulator**, also called a "*dielectric*", is a material that resists the flow of electric current in it. An insulating material has atoms that has tightly bonded valence electrons. These materials are used in parts of electrical equipment, also called *insulators*, with the aim to support or separate electrical conductors without passing current through themselves. The term is also used to refer to insulating supports that attach electric power transmission wires to utility poles or pylons.

Some materials such as glass, paper or Teflon, mica are very good electrical insulators. A much larger class of materials, for example rubber-like materials(polymers) and most plastics are still "good enough" to insulate electrical wiring and cables even though they may have lower bulk resistivity. These materials can act as practical and safe insulators for low to moderate voltages (hundreds, or even thousands, of volts).

Breakdown

Insulators suffer from the phenomenon of electrical breakdown. When the electric field applied across an insulating substance exceeds the threshold breakdown field for that substance, which is proportional to the band gap energy of the insulating material, the insulator suddenly turns into a resistor, sometimes with disastrous results. During electrical breakdown, any free charge carrier being accelerated by the strong e-field will have enough velocity to knock electrons from any atom it strikes. These free electrons and ions are in turn accelerated and strike other atoms, creating more charge carriers, in a chain reaction. Rapidly the insulator becomes filled with mobile carriers, and its resistance drops to a low level. In air, the outbreak of conductivity is called "corona discharge" or a "spark." Similar breakdown can occur within any insulator, even within the bulk solid of a material. Even a vacuum can suffer a sort of break down, but in this case the breakdown or vacuum arc involves charges ejected from the surface of metal electrodes rather than produced by the vacuum itself.

The electrical breakdown of an insulator due to excessive voltage can occur in one of two ways:

- *Puncture voltage:*- is the voltage across the insulator which causes a breakdown and conduction through the interior of the insulator. The heat resulting from the puncture arc usually damages the insulator irreparably.
- *Flashover voltage:*- is the voltage which causes the air around or along the surface of the insulator to break down and conduct, causing a 'flashover' arc along the outside of the insulator. They are usually designed to withstand this without damage.

Most high voltage insulators are designed with a lower flashover voltage than puncture voltage, so they will flashover before they puncture, to avoid damage.

Dirt, pollution, salt, and particularly water on the surface of a high voltage insulator can create a conductive path through it, causing leakage currents and flashovers. The flashover voltage can be more than 50% lower when the insulator is wet. High voltage insulators for outdoor use are shaped to maximize the length of the leakage path along the surface from one end to the other, called the creepage length, to minimize these leakage currents. To accomplish this the surface is molded into a series of corrugations or concentric disk shapes. These usually include one or more *sheds*; downward facing cup-shaped surfaces that act as umbrellas to ensure that the part of the surface leakage path under the 'cup' stays dry in wet weather. Minimum creepage distances are 20–25 mm/kV, but must be increased in high pollution or airborne sea-salt areas.

CLASSES OF INSULTAORS

i) Class-Y insulation: Withstands a temperature of up to 90°C; typically made of cotton, silk, or paper

ii) Class-A insulation: Withstands a temperature of up to 105°C; reinforced Class-Y materials with impregnated varnish or insulation oil

iii) Class-E insulation: Withstands a temperature of up to 120°C

iv) Class-B insulation: Withstands a temperature of up to 130°C. This has a form that inorganic material is hardened with adhesives. This is the first insulator using this structure.

v) Class-F insulation: Withstands a temperature of up to 155°C; for example, made of Class-B materials that are upgraded with adhesives, silicone, and alkyd-resin varnish of higher thermal endurance

vi) Class H insulation: Withstands a temperature of up to 180°C; for example, made of inorganic material glued with silicone resin or adhesives of equivalent performance

vii) Class-C insulation: Withstands a temperature of up to 180°C or higher; made of

100% inorganic material

CHAPTER 4

MODELLING IN MATLAB USING A SET OF TEST RESULTS

Part 1: white minilex paper

Part 2: tamla paper

BREAKDOWN VOLTAGE MODELLING OF ***WHITE MINILEX PAPER*** UNDER AC. CONDITION

Experiment for breakdown of white minilex paper is done varying 3 of its parameters :

- INSULATION THICKNESS
- THICKNESS OF VOID
- DIAMETER OF VOID

By varying these parameters the break down voltage of the insulator is obtained

PARAMETER VARIATION AND CORRESPONDING BREAKDOWN VOLTAGE

Insulation thickness(t)	Thickness of void (t_1)	Diameter of void (d)	Breakdown voltage (V_b)
0.26	0.025	1.5	2.2
0.26	0.125	3	2.2
0.26	0.025	1.5	2.2
0.125	0.125	2	2.3
0.2	0.025	3	2.4
0.18	0.125	3	2.4
0.125	0.025	1.5	2.3
0.125	0.125	1.5	2.2
0.125	0.025	3	2.3
0.125	0.125	5	2.2
0.18	0.025	3	2.2
0.18	0.125	2	2.2
0.18	0.025	3	2.2
0.26	0.125	5	2.2

0.26	0.025	5	2.2
0.26	0.125	2	2.2
0.26	0.025	4	2.2
0.125	0.125	5	2.2
0.18	0.025	2	2.2
0.18	0.125	4	2.2
0.18	0.025	5	2.2
0.125	0.125	5	2.2
0.125	0.025	2	2.3
0.125	0.125	3	2.2
0.18	0.025	5	2.3
0.18	0.125	4	2.4
0.125	0.025	1.5	2.2
0.26	0.125	1.5	2.2
0.18	0.025	2	2.2
0.26	0.125	4	2.2
0.26	0.025	4	2.2
0.26	0.125	1.5	2.3
0.18	0.025	4	2.2
0.26	0.125	4	2.3
0.18	0.025	1.5	2.3
0.18	0.125	4	2.35
0.125	0.025	2	2.2
0.125	0.125	5	2.2

The results obtained are then modeled via ARTIFICIAL NEURAL NETWORK. We use *back propagation error method* to train the model.

MODELLING: IN MATLAB

BACKPROPAGATION ERROR METHOD

Backpropagation, or **propagation of error**, is a common method of teaching artificial neural networks how to perform a given task. (It is a supervised learning method), and is an implementation of the Delta rule. It requires a teacher that knows, the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Back propagation requires that the activation function used by the artificial neurons (or "nodes") is differentiable

Summary

Summary of the backpropagation technique:

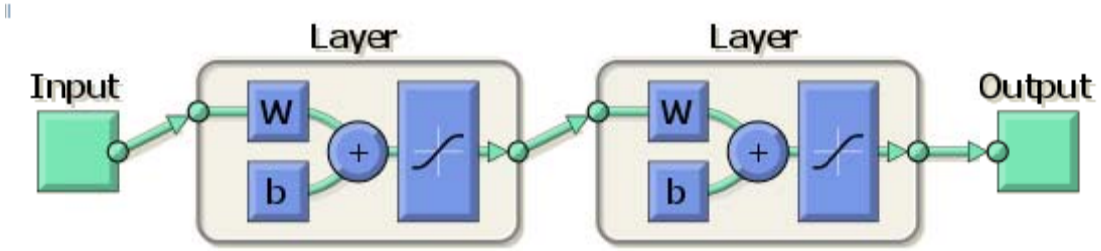
1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. and Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign “output” for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat from step 3 on the neurons at the previous level, using each one's “output” as its error.

ALGORITHM

As the algorithm's name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So, backpropagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights. This gradient is almost always then used in a simple gradient descent algorithm to find weights that minimize the error. Often the term "back propagation" is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in gradient descent. Back propagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

Back propagation networks are necessarily multilayer perceptrons (usually with one input, one hidden, and one output layer). In order for the hidden layer to act as any useful function, multilayer networks must have non-linear activation functions for the multiple layers: a multilayer network using only linear activation functions is equivalent to some single layer, linear network. Non-linear activation functions that are commonly used include the logistic function, the soft max function, and the gaussian function.

The back propagation algorithm for calculating a gradient has been rediscovered a number of times, and is a special case of a more general technique called automatic differentiation in the reverse accumulation mode.



BACK PROPAGATON ERROR: A DETAILED STUDY

1.

Propagates inputs forward in the usual way, i.e.

- All outputs are computed using sigmoid thresholding of the inner product of the corresponding weight and input vectors.
- All outputs at stage *n* are connected to all the inputs at stage *n*+1

2.

Propagates the errors backwards by apportioning them to each unit according to the amount of this error the unit is responsible for.

We now derive the stochastic Backpropagation algorithm for the general case. The derivation is simple, but unfortunately the book-keeping is a little messy.

- $\vec{x}_j =$ input vector for unit j (x_{ji} = ith input to the jth unit)
- $\vec{w}_j =$ weight vector for unit j (w_{ji} = weight on x_{ji})
- $z_j = \vec{w}_j \cdot \vec{x}_j$, the weighted sum of inputs for unit j
- $o_j = \sigma(z_j)$
- o_j = output of unit j ( )
- t_j = target for unit j
- *Downstream*(j) = set of units whose immediate inputs include the output of j

- *Outputs* = set of output units in the final layer

Since we update after each training example, we can simplify the notation somewhat by imagining that the training set consists of exactly one example and so the error can simply be denoted by *E*.

We want to calculate $\frac{\partial E}{\partial w_{ji}}$ for each input weight w_{ji} for each output unit j. Note first that since z_j is a function of w_{ji} regardless of where in the network unit j is located,

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} \\ &= \frac{\partial E}{\partial z_j} x_{ji} \end{aligned}$$

Furthermore, $\frac{\partial E}{\partial z_j}$ is the same regardless of which input weight of unit j we are trying to update. So we denote this quantity by δ_j.

Consider the case when $j \in Outputs$. We know

$$E = \frac{1}{2} \sum_{k \in Outputs} (t_k - \sigma(z_k))^2$$

Since the outputs of all units $k \neq j$ are independent of w_{ji}, we can drop the summation and consider just the contribution to *E* by *j*.

$$\begin{aligned}\delta_j = \frac{\partial E}{\partial z_j} &= \frac{\partial}{\partial z_j}\frac{1}{2}(t_j - o_j))^2 \\ &= -(t_j - o_j)\frac{\partial o_j}{\partial z_j} \\ &= -(t_j - o_j)\frac{\partial}{\partial z_j}\sigma(z_j) \\ &= -(t_j - o_j)(1 - \sigma(z_j))\sigma(z_j) \\ &= -(t_j - o_j)(1 - o_j)o_j\end{aligned}$$

Thus

$$\Delta w_{ji} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta\delta_j x_{ji}$$

Now consider the case when *j* is a hidden unit. Like before, we make the following two important observations.

1. For each unit *k* downstream from *j*, z_k is a function of z_j
2. The contribution to error by all units $l \neq j$ in the same layer as *j* is independent of w_{ji}

We want to calculate $\frac{\partial E}{\partial w_{ji}}$ for each input weight w_{ji} for each hidden unit *j*. Note that w_{ji} influences just z_j which influences o_j which influences $z_k \forall k \in Downstream(j)$ each of which influence *E*. So we can write

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}} &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} \\ &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot x_{ji}\end{aligned}$$

Again note that all the terms except x_{ji} in the above product are the same regardless of which input weight of unit j we are trying to update. Like before, we denote this common quantity by δ_j. Also note that $\frac{\partial E}{\partial z_k} = \delta_k$, $\frac{\partial z_k}{\partial o_j} = w_{kj}$ and $\frac{\partial o_j}{\partial z_j} = o_j(1 - o_j)$. Substituting,

$$\begin{aligned}\delta_j &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \\ &= \sum_{k \in Downstream(j)} \delta_k w_{kj} o_j(1 - o_j)\end{aligned}$$

Thus,

$$\delta_k = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

MODELLING IN NEURAL TOOLBOX

For modeling the breakdown voltage characteristics in solid insulating materials we use the “insulation thickness (t), thickness of void (t1), diameter of void (d)” as input parameters to the ANN. The target parameter is set as “breakdown voltage (Vb)”

We set the input parameters in a matrix [A] and target in matrix [B].

A =

Columns 1 through 9

0.2600	0.2600	0.2600	0.2600	0.1800	0.1250	0.1250	0.1250	0.1250
0.0250	0.1250	0.0250	0.1250	0.0250	0.1250	0.0250	0.1250	0.0250
1.5000	3.0000	1.5000	2.0000	3.0000	3.0000	1.5000	1.5000	3.0000

Columns 10 through 18

0.1800	0.1800	0.1800	0.2600	0.2600	0.2600	0.2600	0.1250	0.1800
0.1250	0.0250	0.1250	0.0250	0.1250	0.0250	0.125	0.0250	0.0125
5.0000	3.0000	2.0000	3.0000	5.0000	5.0000	2.0000	4.0000	5.0000

Columns 19 through 27

0.1800 0.1800 0.1250 0.1250 0.1250 0.1800 0.1800 0.1250 0.2600

0.0250 0.1250 0.0250 0.1250 0.0250 0.1250 0.0250 0.1250 0.0250

2.0000 4.0000 5.0000 5.0000 2.0000 3.0000 5.0000 4.0000 1.5000

Columns 28 through 36

0.1800 0.2600 0.2600 0.2600 0.1800 0.2600 0.180 0.1800 0.1250

0.1250 0.0250 0.1250 0.0250 0.1250 0.0250 0.1250 0.0250 0.1250

1.5000 2.0000 4.0000 4.0000 1.5000 4.0000 4.0000 1.5000 4.0000

Columns 37 through 38

0.1250 0.1250

0.0250 0.1250

2.0000 5.0000

B=

Columns 1 through 9

2.2000 2.2000 2.2000 2.3000 2.4000 2.4000 2.3000 2.2000 2.3000

Columns 10 through 18

2.2000 2.2000 2.2000 2.2000 2.2000 2.2000 2.2000 2.2000 2.2000

Columns 19 through 27

2.2000 2.2000 2.2000 2.2000 2.3000 2.2000 2.3000 2.4000 2.2000

Columns 28 through 36

2.2000 2.2000 2.2000 2.2000 2.3000 2.2000 2.3000 2.3000 2.3500

Columns 37 through 38

2.2000 2.2000

Once the input and the target matrix is set ,we use the *nntool* algorithm to train and simulate the network .

STEPS INVOLVED IN SETTING UP OF THE NEURAL NETWORK

1. In the workspace type *nntool* . This will open a window called the network manager.
2. In the network manager we import the input matrix [A] and the target matrix [B]
3. with this input and target we make a new network
4. we name the network as NETWORK 1
5. NETWORK TYPE : FEED FORWARD BACKDROP
6. TRAINING FUNCTION: *traingd*
7. ADAPTATION LEARNING FUNCTION: *traingdm*
8. PERFORMANCE FUNCTION: *mean square error (mse)*
9. We vary the number of layers to get the best result possible(i.e. least mse)

10. We then train the data for 1000 to 10,000 epochs to train the network till the error in estimation reaches our predesignated value of 0.001 i.e. nearly 1% error in prediction at maximum will be there.
11. We plot the performance graph and the training graph for each case.
12. Compare the trained network results with the experimental results to validate the software. We use the *sim* function to compare the result

```
P=[ t, t1, d];
Sim(net , p)
```

BREAKDOWN VOLTAGE MODELLING OF ***TAMLA PAPER*** UNDER AC. CONDITION

PARAMETER VARIATION AND CORRESPONDING BREAKDOWN VOLTAGE

Insulation thickness(t)	Thickness of void (t1)	Diameter of void (d)	Breakdown voltage (Vb)
0.26	0.025	1.5	0.7
0.26	0.125	3	0.7
0.26	0.025	1.5	0.7
0.125	0.125	2	0.8
0.2	0.025	3	0.7
0.18	0.125	3	0.8
0.125	0.025	1.5	0.9
0.125	0.125	1.5	1
0.125	0.025	3	1
0.125	0.125	5	1
0.18	0.025	3	1
0.18	0.125	2	1
0.18	0.025	3	1
0.26	0.125	5	0.7
0.26	0.025	5	0.9
0.26	0.125	2	0.7
0.26	0.025	4	0.9
0.125	0.125	5	0.7
0.18	0.025	2	1
0.18	0.125	4	1
0.18	0.025	5	0.8
0.125	0.125	5	0.9
0.125	0.025	2	0.9

0.125	0.125	3	0.9
0.18	0.025	5	0.7
0.18	0.125	4	0.8
0.125	0.025	1.5	0.8
0.26	0.125	1.5	0.8
0.18	0.025	2	1
0.26	0.125	4	1
0.26	0.025	4	0.7
0.26	0.125	1.5	0.9
0.18	0.025	4	0.9
0.26	0.125	4	1
0.18	0.025	1.5	0.7
0.18	0.125	4	0.7
0.125	0.025	2	0.7
0.125	0.125	5	0.7

WE USE THE SAME PROCEDURE TO TRAIN AND SIMULATE THE *ANN* FOR *TAMLA PAPER*

HERE THE INPUT MATRIX [A]
A=

Columns 1 through 9

0.2600 0.2600 0.2600 0.2600 0.1800 0.1250 0.1250 0.1250 0.1250

0.0250 0.1250 0.0250 0.1250 0.0250 0.1250 0.0250 0.1250 0.0250

1.5000 3.0000 1.5000 2.0000 3.0000 3.0000 1.5000 1.5000 3.0000

Columns 10 through 18

0.1800 0.1800 0.1800 0.2600 0.2600 0.2600 0.2600 0.1250 0.1800

0.1250 0.0250 0.1250 0.0250 0.1250 0.0250 0.125 0.0250 0.0125

5.0000 3.0000 2.0000 3.0000 5.0000 5.0000 2.0000 4.0000 5.0000

Columns 19 through 27

0.1800	0.1800	0.1250	0.1250	0.1250	0.1800	0.1800	0.1250	0.2600
0.0250	0.1250	0.0250	0.1250	0.0250	0.1250	0.0250	0.1250	0.0250
2.0000	4.0000	5.0000	5.0000	2.0000	3.0000	5.0000	4.0000	1.5000

Columns 28 through 36

0.1800	0.2600	0.2600	0.2600	0.1800	0.2600	0.180	0.1800	0.1250
0.1250	0.0250	0.1250	0.0250	0.1250	0.0250	0.1250	0.0250	0.1250
1.5000	2.0000	4.0000	4.0000	1.5000	4.0000	4.0000	1.5000	4.0000

Columns 37 through 38

0.1250	0.1250
0.0250	0.1250
2.0000	5.0000

TARGET MATRIX [B]

B=

Columns 1 through 9

0.7000 0.7000 0.7000 0.8000 0.7000 0.8000 0.9000 1.0000 1.0000

Columns 10 through 18

1.0000 1.0000 1.0000 0.7000 0.9000 0.7000 0.9000 0.7000 1.0000

Columns 19 through 27

1.0000 0.8000 0.9000 0.9000 0.9000 0.7000 0.8000 0.8000 0.8000

Columns 28 through 36

1.0000 1.0000 0.7000 0.9000 0.9000 1.0000 0.7000 0.7000 0.7000

Columns 37 through 38

0.7000 0.7000

STEPS INVOLVED IN SETTING UP OF THE NEURAL NETWORK

13. In the workspace type *nntool* . This will open a window called the network manager.
14. In the network manager we import the input matrix [A] and the target matrix [B]
15. with this input and target we make a new network
16. we name the network as NETWORK 1
17. NETWORK TYPE : FEED FORWARD BACKDROP

18. TRAINING FUNCTION: *traingd*
19. ADAPTATION LEARNING FUNCTION: *traingdm*
20. PERFORMANCE FUNCTION: *mean square error (mse)*
21. We vary the number of layers to get the best result possible(i.e. least mse)
22. We then train the data for 1000 to 10,000 epochs to train the network till the error in estimation reaches our predesignated value of 0.001 i.e. nearly 1% error in prediction at maximum will be there.
23. We plot the performance graph and the training graph for each case.
24. Compare the trained network results with the experimental results to validate the software. We use the *sim* function to compare the result

```
P=[ t, t1, d];
Sim(net , p)
```

CHAPTER 5

RESULTS AND CONCLUSION
REFERENCES

RESULTS OBTAINED AFTER MODELLING FOR *WHITE MINILEX PAPER*

TEST INPUT [A]
TARGET DATA [B]

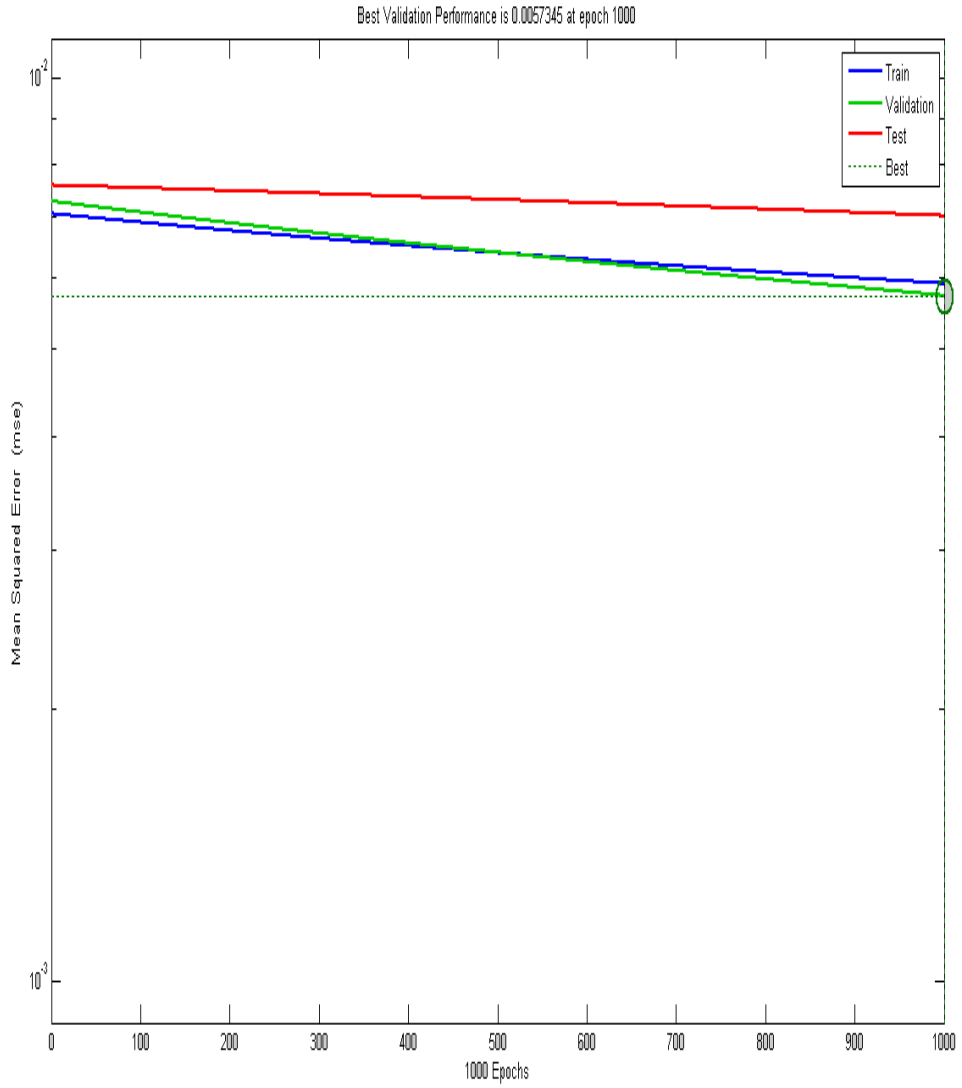
CASE:-1. NO. OF ITERATIONS: 1,000
NO. OF LAYERS: 2

CASE:-2. NO. OF ITERATIONS: 1,000
NO. OF LAYERS: 8

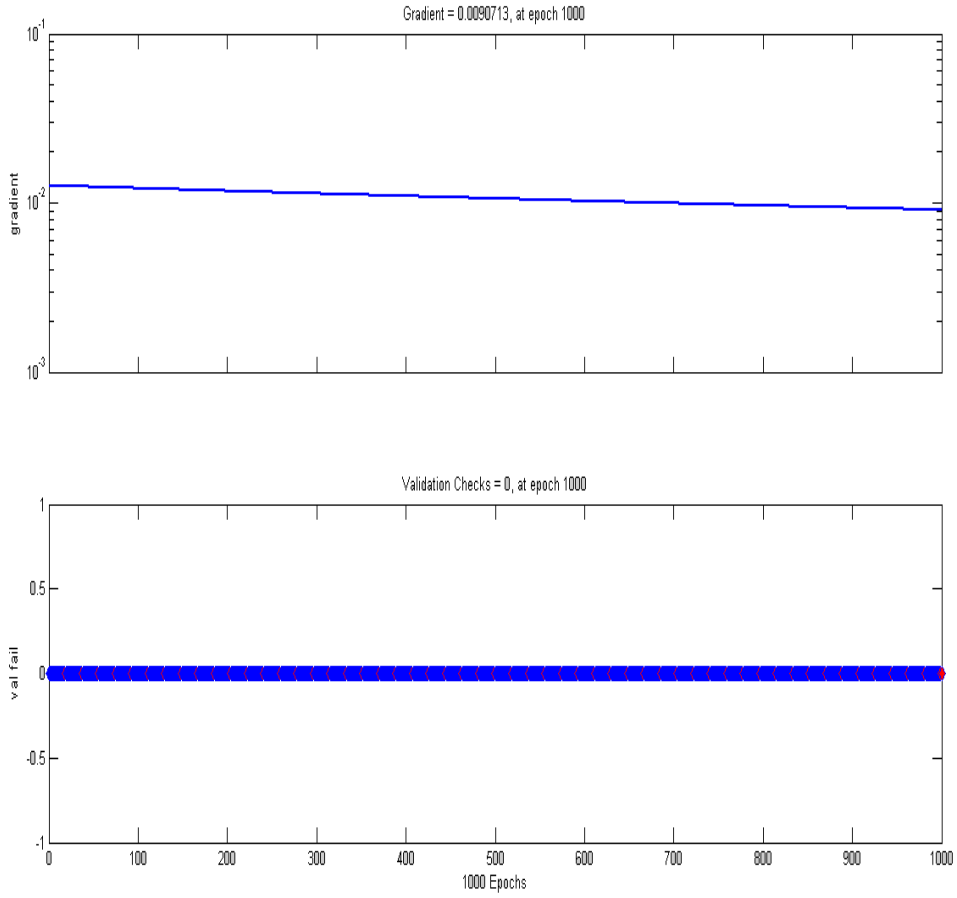
CASE:-3. NO. OF ITERATIONS: 10,000
NO. OF LAYERS: 8

CASE:-1

PERFORMANCE PLOT



TRAINING STATE PLOT

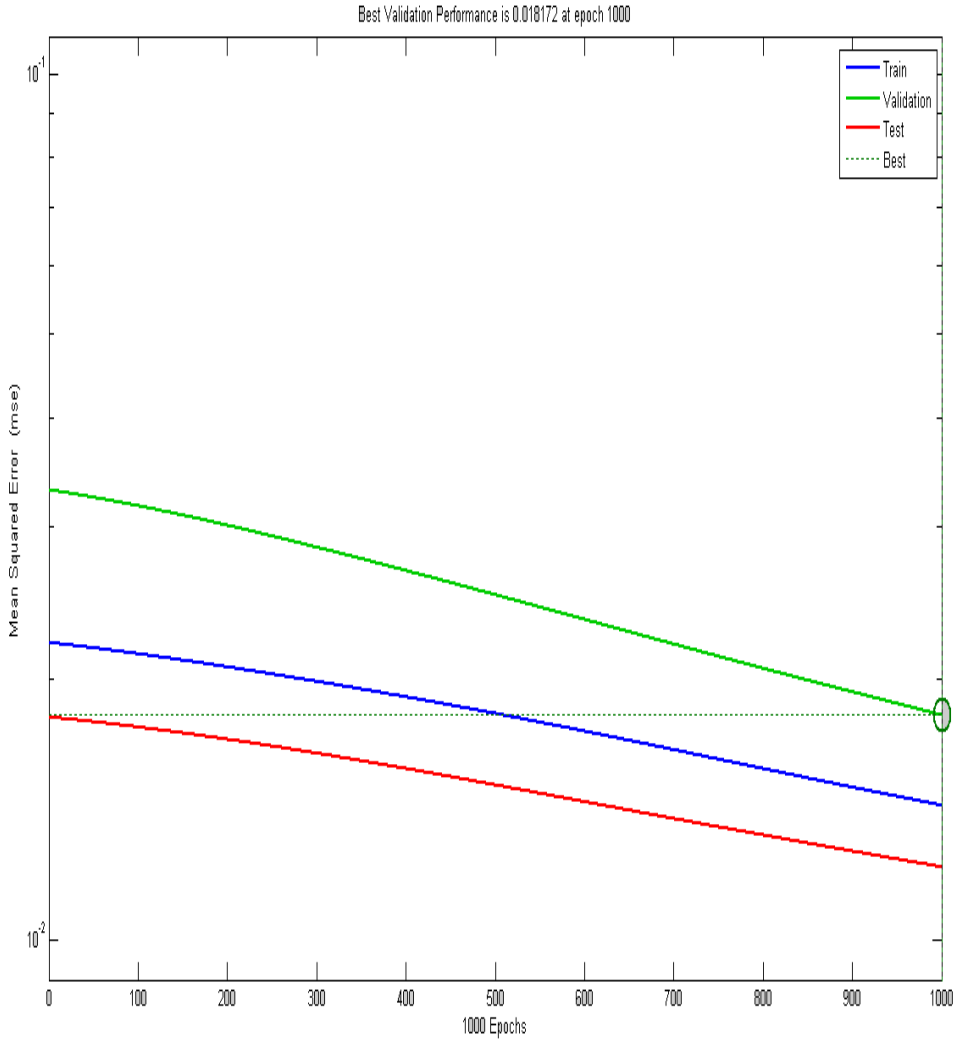


P=[0.27 0.025 3];
sim (net,P)

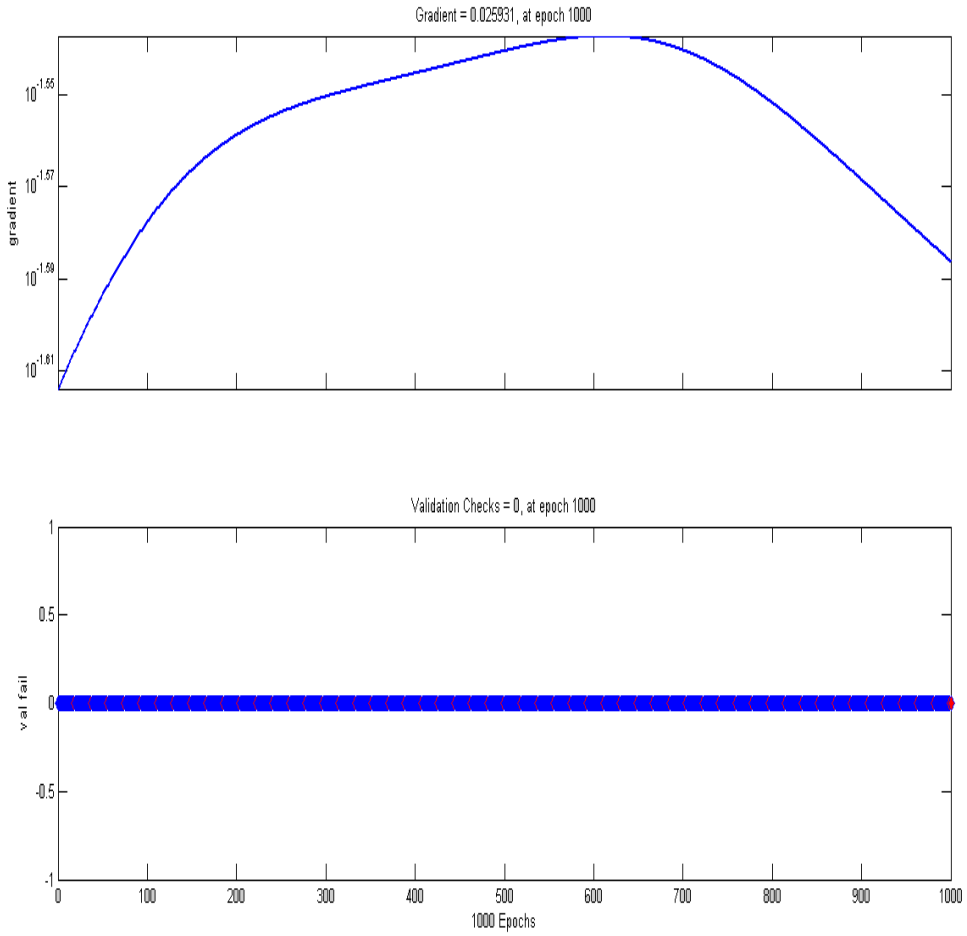
we get the breakdown voltage as 2.2691kv against the measured value of 2.2kv

CASE:-2

PERFORMANCE PLOT



TRAINING STATE PLOT

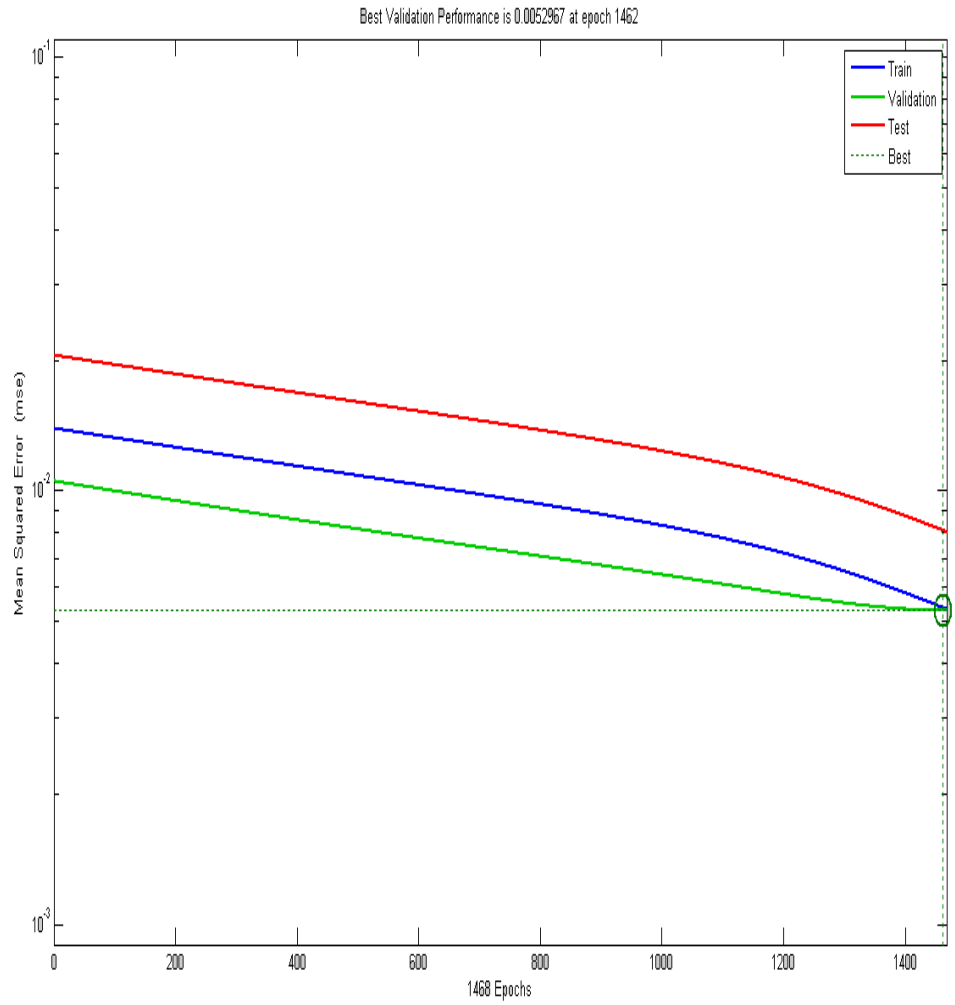


P=[0.27 0.025 3];
sim (net,P)

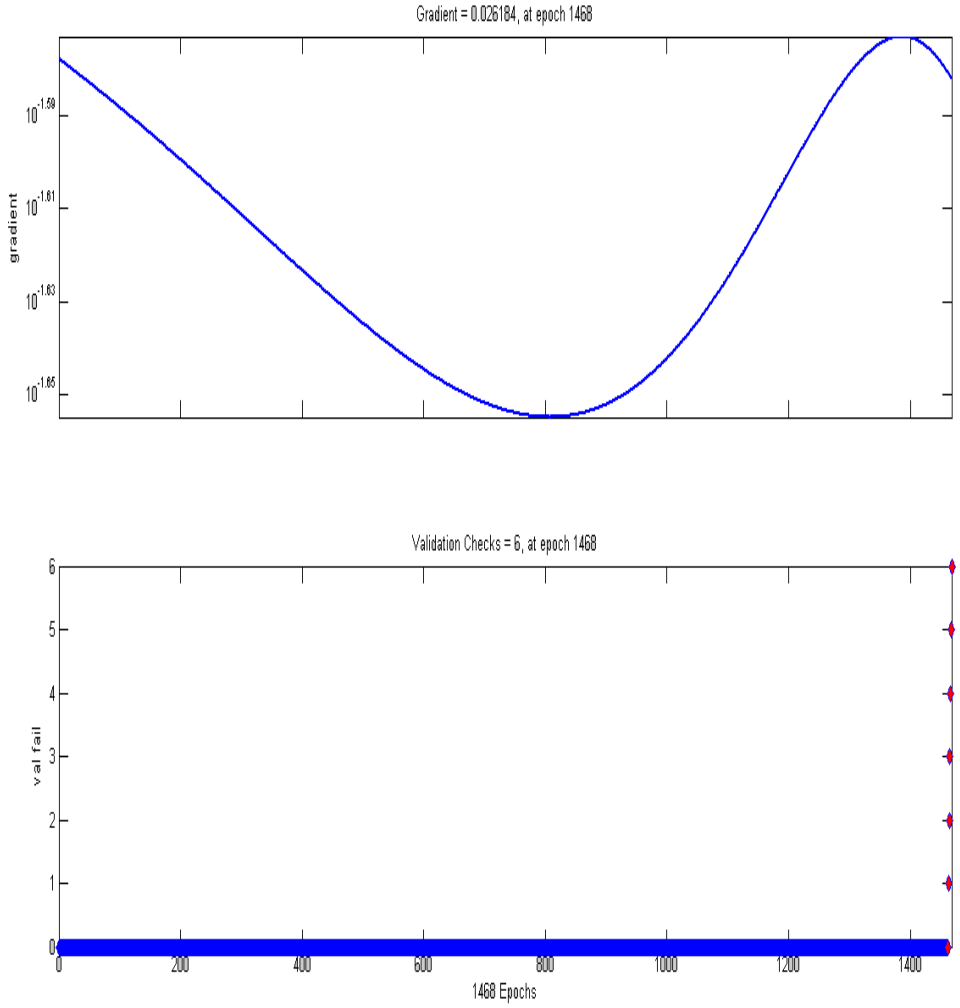
we get the breakdown voltage as 2.2974kv against the measured value of 2.2kv

CASE:-3

PERFORMANCE PLOT



TRAINING STATE PLOT



P=[0.27 0.025 3];
sim (net,P)

we get the breakdown voltage as 2.2399kv against the measured value of 2.2kv

RESULTS OBTAINED AFTER MODELLING FOR *TAMLA PAPER*

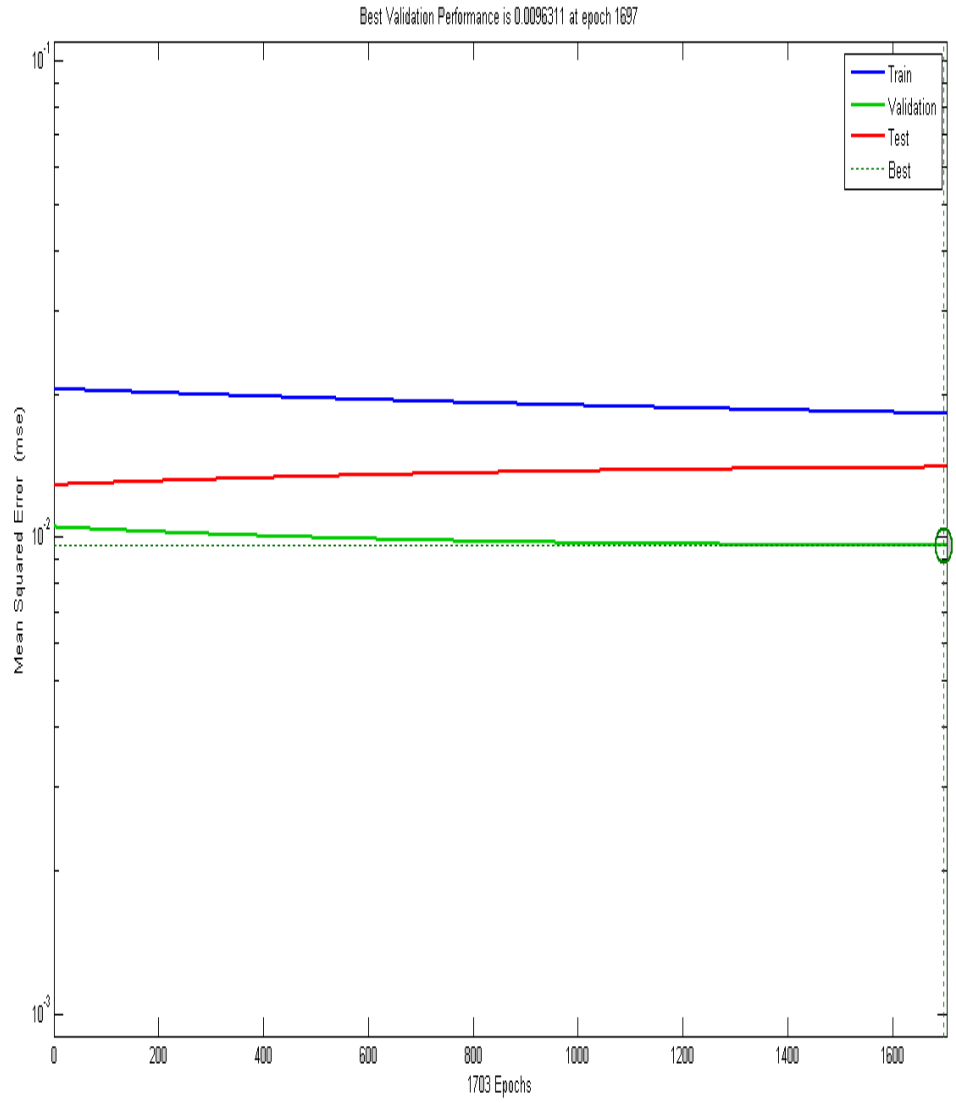
INPUT DATA [A]
TARGET DATA [B]

CASE:-1 NO. OF ITERATIONS: 10,000
NO. OF LAYERS: 2

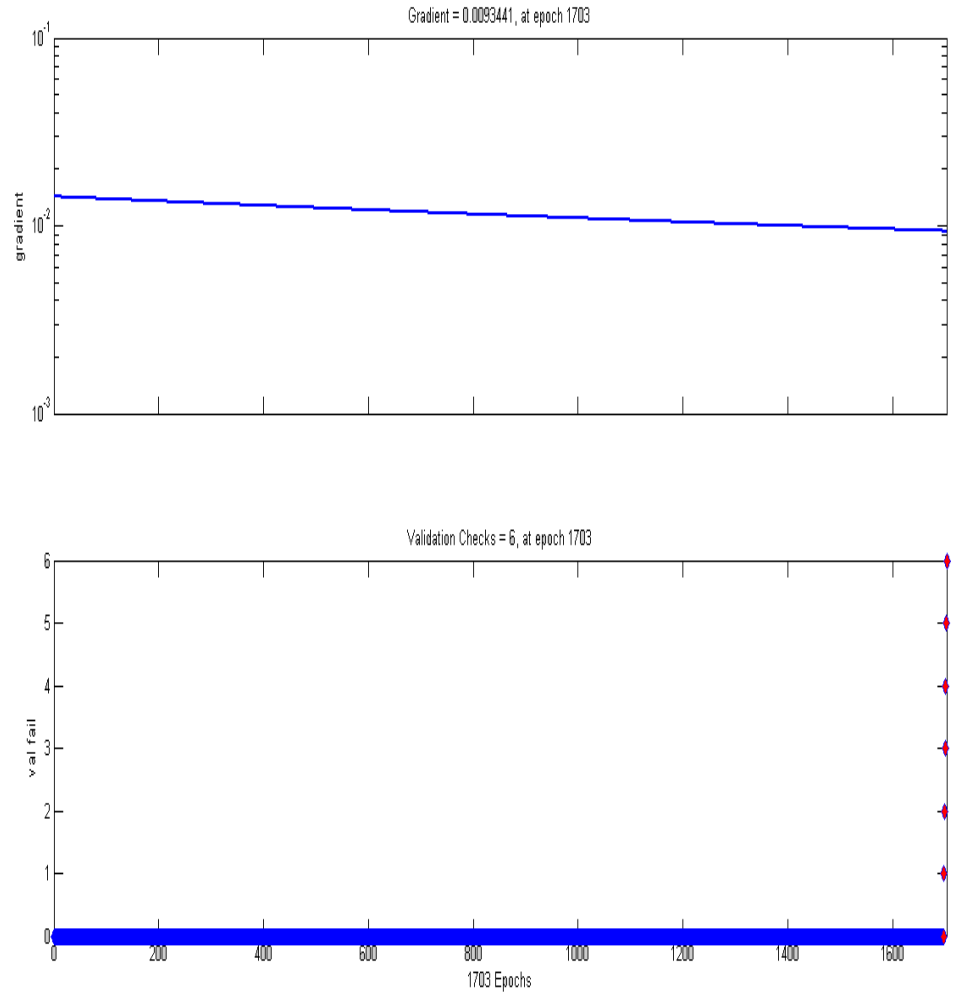
CASE:-2 NO. OF ITERATIONS: 10,000
NO. OF LAYER: 8

CASE:-1

PERFORMANCE PLOT



TRAINING STATE PLOT

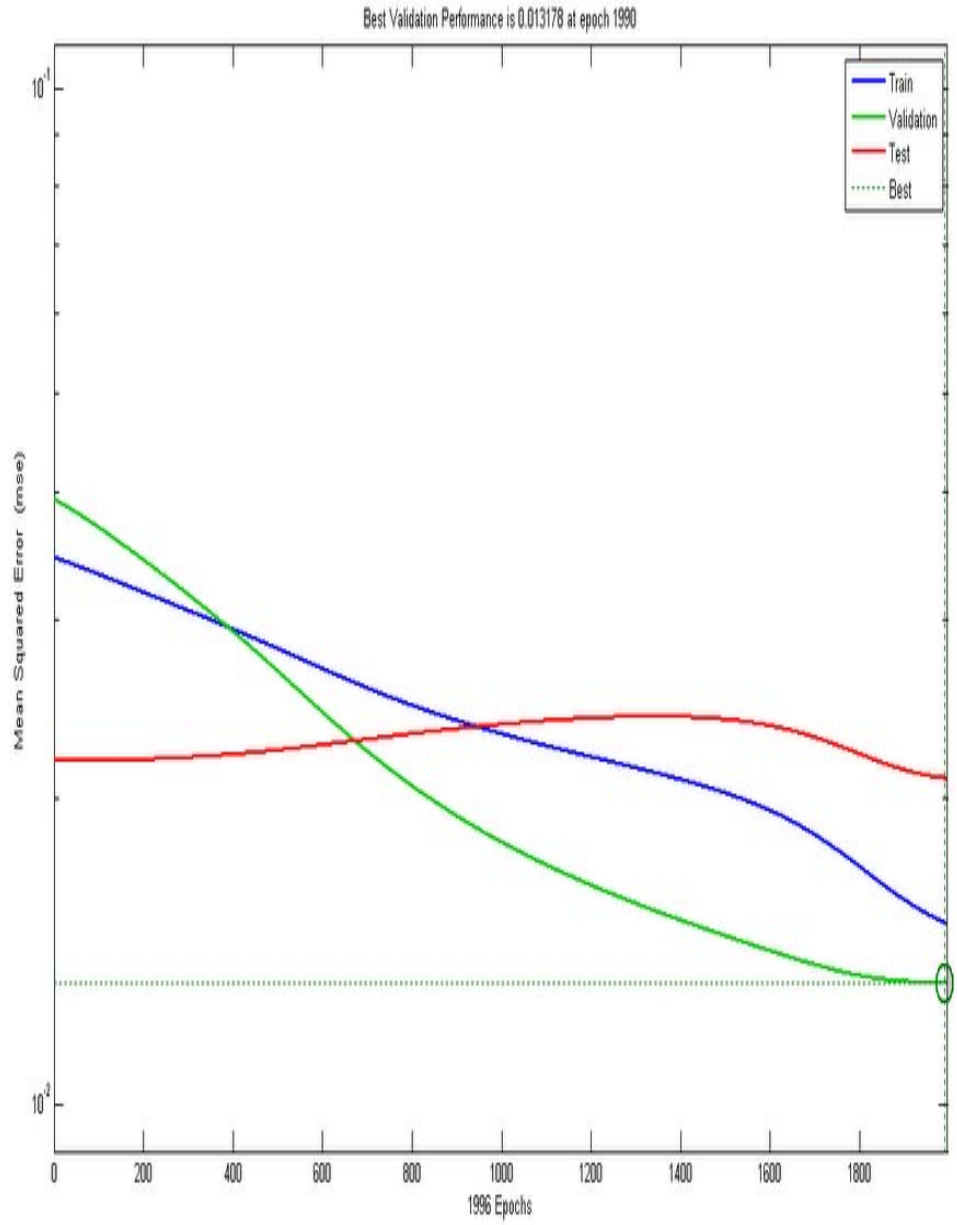


P=[0.27 0.025 3];
sim (net,P)

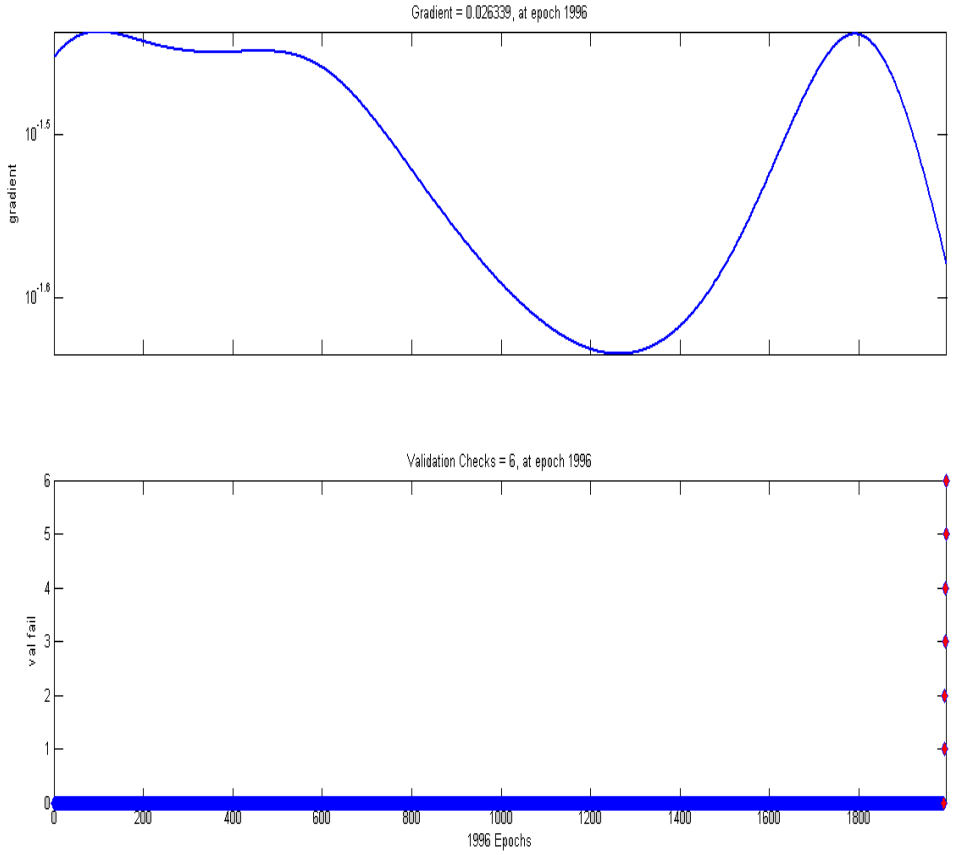
we get the breakdown voltage as 0.8014 kv against the measured value of 0.9kv

CASE:-2

PERFORMANCE PLOT



TRAINING STATE PLOT



P=[0.27 0.025 3];
sim (net,P)

we get the breakdown voltage as 0.8873kv against the measured value of 0.9kv

CONCLUSION

From the above results we conclude that for a ANN model having 8 hidden layers with 3 input variable parameters having the range of 10,000 iterations gives us the best result for the prediction of breakdown voltage of any solid insulating material under normal testing conditions.

Hence 8 hidden layers is suggested for the approximation purpose of estimating the breakdown voltage of solid insulating materials as the error in estimation (*mse*) is very low of the order of 0.001 or 1% error.

We also see that ANN modeling for the modeling of the breakdown characteristics of solid insulating material is a useful way of predicting the breakdown voltage at different condition.

REFERENCES

[1]Fothergill JC. Ageing, Space Charge and Nanodielectrics: Ten things we don't know about dielectrics. Proceedings of IEEE 9th International Conference on Solid Dielectrics, ICSD, Winchester U.K.; 2007: 1-10.

[2] Carton CG. Intrinsic and related forms of breakdown in solids. In: Alston LL, editor. High Voltage Technology, United Kingdom: Oxford University press; 1968, p. 144-183.

[3] Ghosh S, Mohanty S. Modeling of the breakdown voltage of Leatheroid paper in presence of voids using artificial neural networks. Proceedings of IEEE 9th International Conference on Solid Dielectrics, ICSD, Winchester U.K.; 2007: 94-97.

[4] Mohanty S, Ghosh S, Mohapatra SK. Modeling of the breakdown voltage of Leatheroid paper in presence of voids using fuzzy logic techniques. Proceedings of International Conference on Polymeric Materials in Power Engineering, ICPMPE ,Bangalore, India; 2007: 201-210.

[5] Kim SS, Kim MK, Park JK. Consideration of multiple uncertainties for evaluation of available transfer capability using fuzzy continuation power flow. International Journal of Electric Power and Energy Systems. 30(2008) 581-593.

[6] Arya LD, Titare LS, Kothari DP. Probabilistic assessment and preventive control of voltage security margins using artificial neural network. International Journal of Electric Power and Energy Systems. 29(2007) 99-105.

[7] Hammer M, Kozlovsky T, Svoboda J. Fuzzy systems for simulation and prediction of the residual life of insulating materials for electrical

machine windings. Proceedings of IEEE 8th International Conference on Solid Dielectrics, ICSD, France; 2004: 542-545.

[8] Gopal S, Karthikeyan B, Kavitha D. Partial Discharge pattern classification using fuzzy expert system. Proceedings of IEEE 8th International Conference on Solid Dielectrics, ICSD, France; 2004: 500-503.

[9] Hammer M, Kozlovsky T, Svoboda J. The use of neural networks for the life prediction of insulating material of electric rotating machines. Proceedings of IEEE 8th International Conference on Solid Dielectrics, ICSD, France; 2004: 546-549.

[10] Kolev NP, Chalashkanov NM . Modeling of Partial Discharge inception and extinction voltages using adaptive neuro-fuzzy inference system (ANFIS). Proceedings of IEEE 9th International Conference on Solid Dielectrics, ICSD, Winchester U.K.; 2007: 605-608.

[11] ASTM Designation. Standard test methods for sampling and testing untreated paper used for electrical insulation; 2002: D202-97.

[12] Driankov D, Hellendoorn H, Reinfrank M .An introduction to fuzzy control. 2nd ed. Narosa Publishing House, Germany: Springer International Student Union; 1993.

[13] ^ "The Machine Learning Dictionary". http://www.cse.unsw.edu.au/~billw/mldict.html#activnfn.

[14] ^ Roman M. Balabin, Ekaterina I. Lomakina (2009). "Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies". *J. Chem. Phys.* **131** (7): 074104. doi:10.1063/1.3206326.

[15] ^ Izhikevich EM (February 2006). "Polychronization: computation with spikes". *Neural Comput* **18** (2): 245–82. doi:10.1162/089976606775093882. PMID 16378515.

[16] ^ "IBM Research | Press Resources | IBM and EPFL Join Forces to Uncover the Secrets of Cognitive Intelligence".

http://domino.research.ibm.com/comm/pr.nsf/pages/news.20050606_CognitiveIntelligence.html. Retrieved 2009-05-02.
[17]^ B.B. Nasution, A.I. Khan, A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition, IEEE Transactions on Neural Networks, vol 19(2), 212-229, Feb. 2008
[18] ^ Siegelmann, H.T.; Sontag, E.D. (1991). "Turing computability with neural nets". *Appl. Math. Lett.* **4** (6): 77–80. http://www.math.rutgers.edu/~sontag/FTP_DIR/aml-turing.pdf.
[19] www.google.com
[20] www.wikipedia.org